

Making Canonical Workflow Building Blocks Interoperable across Workflow Languages

Stian Soiland-Reyes^{1,2†}, Genís Bayarri³, Pau Andrio⁴, Robin Long^{5,6}, Douglas Lowe⁶,
Ania Niewielska⁷, Adam Hospital³ & Paul Groth²

¹Department of Computer Science, The University of Manchester, Manchester M13 9PL, UK

²Informatics Institute, University of Amsterdam, Amsterdam 1098 XH, The Netherlands

³Institute for Research in Biomedicine (IRB Barcelona), The Barcelona Institute of Science and Technology (BIST), Barcelona 08028, Spain

⁴The Spanish National Bioinformatics Institute (INB), Barcelona Supercomputing Center (BSC), Barcelona 08034, Spain

⁵Data Science Institute, Lancaster University, Lancaster, Lancashire LA1 4YW, UK

⁶Research IT, IT Services, The University of Manchester, Manchester M13 9PL, UK

⁷European Bioinformatics Institute (EMBL-EBI), Hinxton, Cambridgeshire CB10 1SD, UK

chinaXiv:202211.00436v1

Keywords: Scientific workflows; Interoperable; FAIR; Computational tools; Containers; Software packaging; FAIR digital object (FDO), BioExcel Building Blocks library (BioBB), Canonical Workflow Frameworks for Research (CWFR)

Citation: Soiland-Reyes, S., et al.: Making canonical workflow building blocks interoperable across workflow languages. Data Intelligence 4(2), 342-357 (2022). doi: 10.1162/dint_a_00135

Received: October 1, 2021; Revised: November 26, 2021; Accepted: February 5, 2022

ABSTRACT

We introduce the concept of *Canonical Workflow Building Blocks* (CWBB), a methodology of describing and wrapping computational tools, in order for them to be utilised in a reproducible manner from multiple workflow languages and execution platforms. The concept is implemented and demonstrated with the BioExcel Building Blocks library (BioBB), a collection of tool wrappers in the field of computational biomolecular simulation. Interoperability across different workflow languages is showcased through a protein Molecular Dynamics setup transversal workflow, built using this library and run with 5 different Workflow Manager Systems (WfMS). We argue such practice is a necessary requirement for FAIR Computational Workflows and an element of Canonical Workflow Frameworks for Research (CWFR) in order to improve widespread adoption and reuse of computational methods across workflow language barriers.

[†] Corresponding author: Stian Soiland-Reyes (Email: soiland-reyes@manchester.ac.uk; ORCID: 0000-0001-9842-9718).

1. INTRODUCTION

The need for *reproducibility* of research software usage is well established [1, 2, 3], and adaptation of *workflow management systems* (WfMS) together with *software packaging and containers* [4] have been proposed as key ingredients for making research software usage *FAIR* and reproducible [5, 6, 7]. Recently it is also argued that computational workflows should also be treated as FAIR Digital Objects [8] in their own right, with identifier, metadata [2] and interoperability requirements [9].

[BioExcel](#), a European Centre of Excellence for Computational Biomolecular Research, has a particular focus on the research domains molecular dynamics simulations and bioinformatics with use of *High Performance Computing* (HPC) to approach Exascale performance, while also improving usability. The *BioExcel Building Blocks* (BioBB) [10] have been created as portable wrappers of open-source computational tools identified as useful for BioExcel workflows, forming several *families* of documented and interoperable operations that can be called from multiple workflow systems. This interoperability is shown with the BioBB demonstrator workflows, along with multiple tutorials and notebooks.

We propose that these building blocks and their families themselves can be considered *composite Digital Objects*: collections of software packages and their source code, guides and tutorials, as well as workflow management system integrations and workflow examples. In addition, the building blocks, as wrappers of upstream open source tools, benefit from and refer to the tools' existing documentation, support forums, academic publications and wider development context.

Given BioBB as a starting point, we define a generalized methodology of *Canonical Workflow Building Blocks (CWBB)*, through the definition of a set of requirements and recommendations for how to formalize and develop a family of compatible computational tools as Digital Objects. These building blocks let researchers instantiate a Canonical Workflow in multiple workflow management systems, while also benefiting from the FAIR aspects of the CWBB Digital Objects.

2. METHODS

The [BioExcel Building Blocks](#) library [10], created and implemented within the BioExcel CoE, is a collection of portable wrappers of common biomolecular simulation tools. The BioBB library is designed to i) increase the *interoperability* between the tools wrapped; ii) ease the *implementation* of biomolecular simulation workflows; and iii) increase the *reusability and reproducibility* of the generated workflows. To achieve these main goals, the library was designed following the FAIR principles for research software development best practices [7].

The result is a collection of building block modules, divided in sets of tool wrappers focused on similar functionalities (e.g. Molecular Dynamics, Virtual Screening). Each of the modules is built from a combination of (i) software packaging ([Pip](#), [BioConda](#), [BioContainers](#)), (ii) documentation ([ReadTheDocs](#)), (iii) interactive tutorials ([Jupyter Notebooks](#), [Binder](#)), (iv) registry & findability ([bio.tools](#), [BioSchemas](#), [WorkflowHub](#)), (v)

WfMS integration stubs ([CWL](#), [Galaxy](#), [PyCOMPSs](#)), (vi) source Code ([GitHub](#)) and (vii) REST APIs ([OpenAPI](#), [Swagger](#)). Notably all building blocks follow the same pattern of installation, configuration and interaction.

Since the publication of the library, several [new building block modules](#) (Chemistry, Machine Learning, AMBER MD, Virtual Screening, etc.) have been added, and the set of operations for the existing BioBB families have been expanded. While we previously provided curated adapters (Figure 1) for running BioBB in workflow systems using Common Workflow Language (CWL) and PyCOMPSs, along with Galaxy Toolshed bindings, we have now started auto-generating these bindings, along with command line wrappers and REST web service APIs, using annotations within BioBB's Python docstrings as source. These annotations include sufficient information for a WfMS to launch a particular building block: *input* and *output* parameters (including *mandatory*/*optional* flags), compatible *formats* (including from EDAM ontology [11]), *example* files (essential for testing purposes), default values and *dependencies*. This ensures human-readable documentation, FAIR metadata and programmatic accessibility can be generated consistently and comparably.

```

1 #!/usr/bin/env cwl-runner
2 cwlVersion: v1.0
3
4 class: CommandLineTool
5
6 label: Wrapper class for the GROMACS editconf module.
7
8 doc: |
9   The GROMACS solvate module generates a box around the selected structure.
10
11 baseCommand: editconf
12
13 hints:
14   DockerRequirement:
15     dockerPull: https://quay.io/biocontainers/biobbb_md:3.6.0--pyhdf678af_0
16
17 inputs:
18   input_gro_path:
19     label: Path to the input GRO file
20     doc: |->
21       Path to the input GRO file
22     type: string
23     file_type: Input
24     Accepted formats: gro
25     Example file: https://github.com/bioexcel/biobbb_md/raw/master/biobbb_md/test/data/gromacs/editconf.gro
26 type: File
27 forward:
28 - edamFormat: 2033
29 inputsBinding:
30   positions: 1
31   prefix: --input_gro_path
32
33
34 <tool id="biobbb_md_editconf_ext" name="Editconf" version="3.0.0" >
35   <description>Wrapper Class for the GROMACS editconf module.</description>
36   <requirements>
37     <requirement type="package">biobbb_md</requirement>
38   </requirements>
39   <command detect_errors="aggressive"><![CDATA[
40
41     In -f -s ${input_gro_path} ${input_gro_path}.ext;
42
43     If $config.sel == "option1":
44       In -s -f ${config.properties} ${config.properties}.ext;
45     End if
46
47   editconf;
48
49   If $config.sel == "option1":
50     --config ${config.properties}.${config.properties.ext};
51   Else if $config.sel == "option2":
52     --config ${config.jsonstr};
53   Else if $config.sel == "option3":
54     --config " -oc _dq_distance_to_molecule_dq_ -${config.distance_to_molecule},
55             _dq_box_type_dq_ -_dq_${config.box_type}_dq_ -_dq_center_molecule_dq_ -${config.
56             center_molecule}, _dq_gmxlib_dq_ -_dq_${config.gmxlib}_dq_ -_dq_gmx_path_dq_ -_dq_${config.
57             gmx_path}, _dq_new_file_path_dq_ -#outname_output_gro_path"
58   End if
59
60   If str(${input_gro_path}) != 'None':
61     --input_gro_path ${input_gro_path}.${input_gro_path}.ext
62   End if
63
64   --output_gro_path ${_new_file_path}_#outname_output_gro_path
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
247
248
249
249
250
251
252
253
254
255
256
257
257
258
259
259
260
261
262
263
264
265
266
267
267
268
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
145
```

The library is showcased through a collection of [demonstration workflows](#) [12]. Here, each workflow introduces individual building blocks as needed to explain a particular scientific computational method. We primarily expose the workflows as Jupyter Notebooks [13], which has been highlighted as a valuable tool for reproducible scientific workflows [14]. This offers a graphical interactive interface, including documentation (integrated markdown) related to the workflow and the building blocks used, but also to the biomolecular simulation methods used in the pipeline. Moreover, as we have demonstrated with our own [Binder](#) [15] hosting, these workflows are reproducible across platforms, assisted by BioConda [16] packaging of the building blocks and their software dependencies.

This assembly of available demonstration workflows has been successfully used in the BioExcel CoE for dissemination with a range of [training events](#) (e.g., BioExcel Summer & Winter School, webinars and virtual training). In training we particularly utilised the Binder infrastructure of the BioExcel Cloud portal [17] to give users a web-based first experience of the building blocks before they try them in other workflow systems.

We can observe that workflow building blocks such as BioBB are necessarily composed of a comprehensive list of digital objects, encompassing source code, packaging, containerization, documentation, attributions, citations, registry entries, WfMS integrations and REST APIs.

We propose to consider building blocks as *composite digital objects* in their own right: gathering the above software components along with their metadata, identifiers and operations then forms a *Canonical Workflow Building Block* (CWBB). We suggest this concept as a fundamental element of FAIR Digital Objects for Computational Workflows: researchers use the building blocks computationally as functional operations across WfMSs, while the FAIR aspect of CWBB propagates information and resources that are essential for reproducibility, reuse and understanding by anyone discovering the workflow.

2.1 Interoperability across Different Workflow Languages

The concept of Canonical Workflow Building Blocks is here showcased with the BioBB library, by using a transversal workflow present in many different computational biomolecular projects: a [Molecular Dynamics \(MD\) protein setup](#). This workflow prepares a protein structure to be used as input for an MD simulation, going through a series of steps where the protein is completed (adding hydrogen and missing atoms), optionally introducing a residue mutation, then submerging the protein in a virtual box of water molecules with a particular ionic concentration, and finally energetically equilibrating the system (so that solvent and ions are well accommodated around the protein at the desired temperature).

This simulation process involves a non-negligible number of steps, using a variety of biomolecular tools. The BioBB library was used to assemble this workflow, interconnecting building blocks using Python functions (Jupyter Notebook, Command Line Interface), auto-generated bindings (Galaxy [18], CWL [19], PyCOMPSs [20]) or manually generated bindings (KNIME [21]). Corresponding workflows for the different WfMS can be found in [WorkflowHub](#) [22, 23, 24, 25, 26] and graphical extracts can be seen in Figure 2.

chinaXiv:202211.00436v1

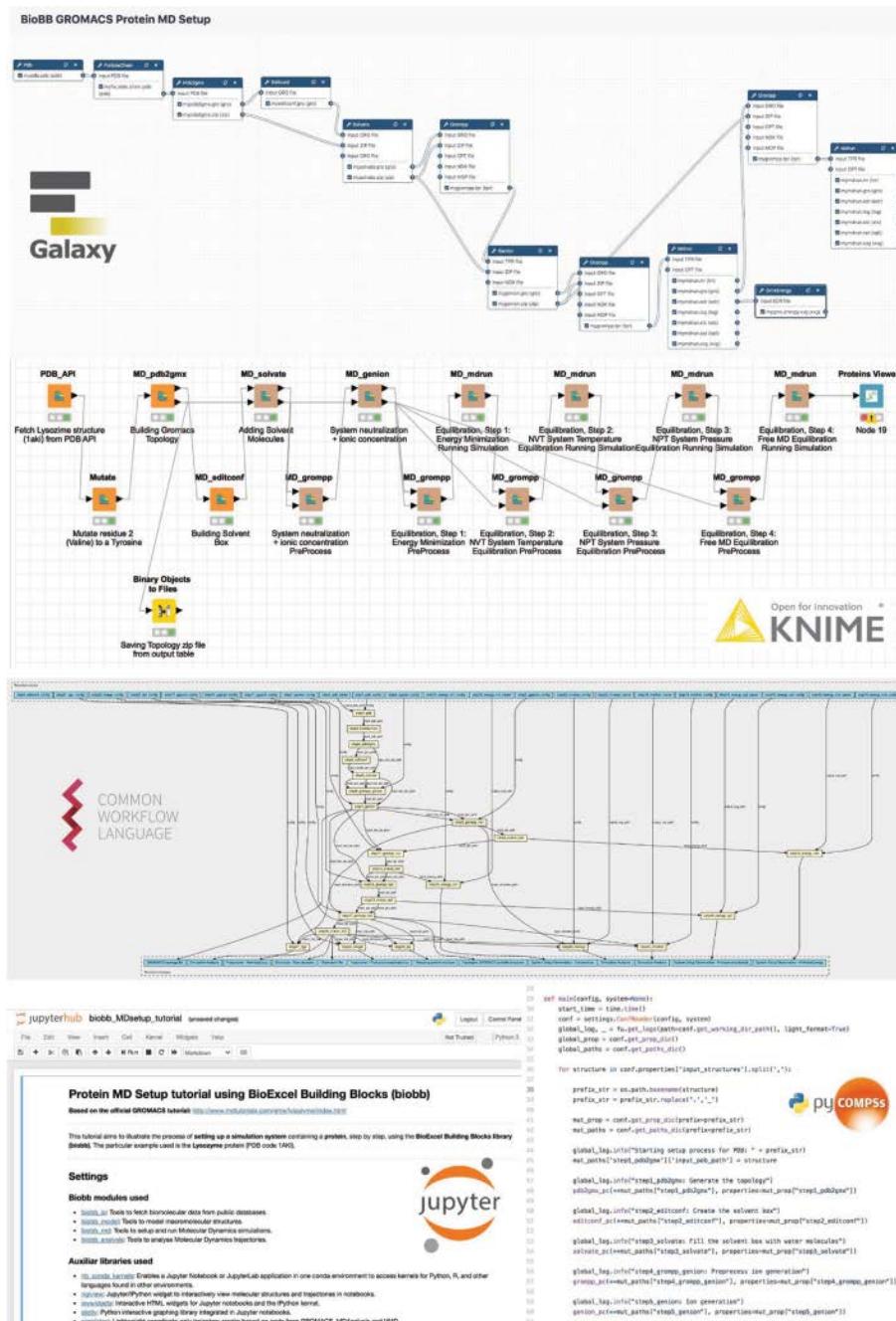


Figure 2. Protein MD Setup transversal workflow, assembled in with 5 different workflow managers using BioBB canonical building blocks. From top-left: Galaxy [22], KNIME [23], CWL [24], Jupyter Notebook [25], and PyCOMPSs [26].

This example demonstrates how the same canonical building blocks can be used in different WfMS. Wrappers and tools executed behind the workflows are exactly the same, but the workflows are built using different WfMS, some of them in a graphical way (Galaxy, KNIME), some in a command line way (Jupyter Notebook, PyCOMPSs, CWL); workflows can be focused on short/interactive executions (Jupyter Notebook), or on High Throughput/High Performance Computing (HT-HPC) executions (PyCOMPSs); some of them prepared for a particular WfMS installation (Galaxy), others completely system-agnostic (CWL).

The current WfMS bindings include Jupyter Notebook, PyCOMPSs, CWL, Galaxy and KNIME WfMS, in addition to a [command line](#) mechanism. Thanks to the extensive documentation added in the source code as Python docstrings, new bindings for available WfMS can be generated. We are also experimenting with generating a REST API exposing the building services as Web services. However, it should be noted that such automatic generation of bindings is not always practically feasible. As an example, KNIME nodes require a complete Java skeleton code, as well as a definition of new data types for all inputs/outputs required, which makes their automatic generation a heavy and potentially error-prone task. Bindings for workflow languages with a *domain-specific language* (DSL) for tool definitions (e.g. Galaxy, CWL) can, on the other hand, be generated in a more straightforward fashion.

The transversal [protein MD setup workflow](#) was chosen as a real example that is readily understandable by domain experts. More [complex pipelines](#) involving a broader set of wrapped biomolecular tools have been developed using the BioBB library, primarily as Jupyter Notebooks. A selection of these have similarly been assembled for different WfMS using the auto-generated bindings and uploaded to the [WorkflowHub repository](#).

3. DISCUSSION

Early work on libraries of workflows fragments include Web Service-based approaches where tools are wrapped and exposed using common, interoperable data types in [BioMoby](#) [27] for bioinformatics and similarly [caBIG](#) [28] for cancer genomics. While these efforts were interoperable across WfMSs they required a large up-front investment in agreeing to and adapting native data to common RDF or XML representations.

The notion of *abstract workflows* [29], structural workflow descriptions separated from their concrete execution realisations and augmented with Linked Data annotations, have been emphasised as essential for reuse and consistency across workflow systems. Identifying *common motifs* for workflow operations [30] (e.g. Data preparation, Format transformation, Filter, Combine) are important to simplify and understand otherwise fine-grained workflow provenance traces.

Most other efforts to standardise a set of disparate analytical tools have been done within the scope of a single WfMS, allowing customised user interaction, data visualisation, configuration and findability, for instance [Taverna components](#) had prototypical building blocks [31] which were instantiated at runtime by reference from a registry. [KNIME components and metanodes](#), shared on the [KNIME Hub](#) are frequently designed

to be interoperable, but with a perhaps weaker notion of component families. The [Galaxy toolshed](#) [32] is likewise populated with different sets of tool wrappers that are largely made to be interoperable within a category.

The *Common Workflow Language* (CWL) [19] has a strong emphasis on interoperable command line tool descriptions, with [support for containers](#) and Conda packaging, as well as [support for FAIR metadata](#) like contributors, license and EDAM ontology type annotations. With multiple leading workflow engines now supporting CWL, and experimental Galaxy support, this seems perhaps the most promising candidate for both making and describing canonical workflow building blocks, however we've identified a few stumbling blocks.

One obvious challenge is that the implementing WfMS needs to have CWL support, along with support for either containers or Conda packaging to find the described executables. While it is possible to run a CWL tool directly using a `#!/usr/bin/env cwl-runner` shebang on POSIX systems, this still requires pre-installation and possibly configuration of a CWL engine like [cwltool](#) or [Toil](#) [33]. However workflow engines have multiple dependencies and often cannot easily be run from a container themselves^①.

Within the CWL community it was originally envisioned that a wider set of workflow systems would adopt CWL for tool description/execution, with a subset implementing full CWL workflow support. This would allow shared community effort for describing tools, say in the [Common Workflow Library](#), rather than each WfMS needing to duplicate this tool wrapping in separate repositories and languages. However, with the exception of experimental tool support in Galaxy, in practice all CWL implementers have gone for full workflow support.

Another challenge is that making a set of building blocks frequently requires the use of *shims*, for instance file conversion, small search/replace operations or file renames. In a CWL approach these can either be performed with an [Expression](#) using JavaScript snippets which only has limited access to file content, or as an additional workflow step added before or after the main tool step. This combination could then be nested as a subworkflow, similar to KNIME's *metanodes*, and would also be flexible by allowing different containers or packages for any pre- or post-steps. Such a CWL building block however becomes harder to access from a non-CWL WfMS, because of lack of control over configuration/execution options for the now nested CWL tools. In practice^②, executing a nested CWL workflow from a native WfMS language would require the engine to implement full CWL Workflow support (or delegate to a CWL engine).

For the main BioBB building blocks we implemented [demonstrator workflows](#) that highlight how the tools should be used in different workflow management systems; each having a primary exemplar using Jupyter Notebook, which can be explored interactively using the [BioExcel Binder](#). If we consider the abstract

^① To execute the wrapped tool, a containerized workflow engine would need *nested containers* which are not generally recommended for security reasons. It is possible to work around this limitation using [Singularity](#) or [Conda](#).

^② It is worth mentioning that it would also be possible to generate WfMS-specific bindings from CWL descriptions (e.g. as demonstrated with [cwl2script](#) for Bash, [gxargparse](#) for Galaxy, [cwl2wdl](#) for WDL), although this necessitates constraining the tool and workflow definitions to a limited mappable subset of CWL.

demonstrator workflows as *canonical workflows* they are therefore very much active objects, but can also be seen as *workflow templates*, as any real use case will need to specialise the workflow to tweak parameters, data selection etc.

We therefore also provide such workflow templates for multiple WfMS, including CWL, PyCOMPSs and Galaxy. These are fairly disparate workflow languages, yet by the use of the same canonical workflow building blocks (which again invoke the same software binaries), such WfMS-specific workflows effectively are instantiations of the same canonical workflow.

One challenge found is how to publish such canonical workflows in registries like the [WorkflowHub](#). The hub supports the registration of Digital Objects in the form of RO-Crate [34], with the option of abstract CWL for describing the canonical workflow template, along with direct references to the workflow's GitHub repository.

For instance in the RO-Crate for <https://doi.org/10.48546/workflowhub.workflow.200.1> [26], which can also be [rendered](#) from GitHub, we have an entry for the [main workflow](#) according to the [Workflow RO-Crate profile](#), detailing each canonical workflow building block used (e.g., [biobb-md metadata](#)). Here the FAIR aspect of the building blocks to help software citation is exercised, as the building block wrapper has one set of authors, documentation and licence (Apache-2.0), while the wrapped software (e.g. [GROMACS metadata](#)) has different authors, licence (GPL-2.1+) and documentation.

However, the deposit of such RO-Crates in WorkflowHub results in one registration entry per workflow language, which are not otherwise related and may not even share the same source code repository. Thus, we've identified the need for adding an overall *canonical workflow entry*, which can bring in workflow documentation and references shared across WfMS implementations, including a set of links to the more granular canonical workflow building blocks used by the workflow, but also to the individual WfMS implementations as separate digital objects.

A similar question of granularity applies at the workflow tool level [4], particularly for Findability and Accessibility, as we can consider at lowest granularity the *scientific method* in general (e.g., any algorithm for sequence alignment), followed by an *application suite* (bio.tools entry [35], homepage, documentation), instantiated as a particular *software installation* (Debian package, Docker container) with its dependencies at same level. The installation includes one or more *software executables* (a particular binary, a running service), providing at the highest detailed granularity level the specific types of *software functionality* (a particular mode of operation, choice of analysis), for instance using certain command line flags.

For canonical workflow building blocks, with a focus on pluggable composability, this is mainly defined at this high granularity level of specific software functionality: explicit operations from an installed tool, which are then combined in a workflow. This is indeed the level WfMS tool definitions are typically done, e.g., a CWL Command Line Tool specifies a particular way to run a particular software binary. However, to be an actionable CWBB, the building block needs to additionally convey the lower granularity levels; particularly to support multiple options for interoperable installation and execution, as well as metadata at the most general level, such as documentation and scholarly citations.

While workflow management systems typically only operate at the highest granularity levels for execution details, and are frequently unaware of (or not exposing metadata at) the more general levels, we argue that in order for a Canonical Workflow [36] to follow and support FAIR principles for itself and its data, the workflow management system needs to *propagate structured metadata* about the tools used by the workflow. We propose that in order to support the workflow's applicability to multiple WfMS, the tools themselves must also have a consistent packaging and formal description that enables computational invocation.

At the most general level, a canonical workflow built using such CWBBs is even conceptually reproducible because the FAIR documentation of the workflow, through its canonical workflow building blocks, identifies how individual tools and software applications are composed, which in worst case can be rebuilt using different installation methods in a different WfMS, or in best case inspected to detect and cross-link the same canonical workflow appearing in different WfMS instantiations. This view of software as composition of other software typically also applies at individual tool level, which themselves depend on programming language runtimes, libraries, services and reference data.

4. REQUIREMENTS FOR CANONICAL WORKFLOW BUILDING BLOCKS

Building on the experiences with BioBB, we here propose requirements and recommendations for establishing Canonical Workflow Building Blocks (CWBB) as implementations of *canonical steps* introduced for Canonical Workflow Frameworks for Research [36].

The core purpose of a CWBB is to wrap a command line tool or other software that can perform an operation as part of a computational workflow. As such, the general advice for making software workflow-ready applies [37] (e.g., easy to install, documented, parallelizable, reproducible output), however a CWBB is also permitted to make use of additional scripts or *shims* to further adapt a third-party tool for workflow use and for data interoperability across blocks.

The way tools are installed or invoked varies slightly across WfMS and operating systems, therefore a CWBB should provide multiple methods for distributing software; currently containers (Docker, Singularity) and distribution-independent packaging (e.g., Conda, Homebrew) are promising by having reproducible install recipes and a wide range of available open source dependencies (e.g. Java, Python). Additionally building blocks should allow overriding execution paths, e.g., for use with HPC module system and hardware-optimised binaries.

The CWBBs should have sufficient annotations to be able to generate bindings for different WfMSs and REST APIs, e.g. parameter names and descriptions, types and default values; enumerators for options, file formats for inputs/outputs.

Building blocks should be grouped into families that are interoperable through common data structures and file formats, as well as having joint naming conventions for configuration options. A CWBB family should be released as a single version following [semantic versioning](#) rules, which should have a corresponding persistent identifier (PID) [38].

Metadata for CWBBs should be captured following FAIR guidelines, and distributed as part of the block family and resolvable from the PID as a FAIR Digital Object. Metadata should include references to the CWBB software distributions (e.g., [quay.io](#) container URL) as well as attributions, citations and documentation for the wrapped tool.

Example workflows showing CWBB usage should be included in a WfMS-neutral language such as Jupyter Notebooks, which may have equivalent variants for each workflow binding. These workflows should be registered in a workflow registry like WorkflowHub or Dockstore, and assigned their own PIDs.

5. CONCLUSIONS

The proposed concept of Canonical Workflow Building Blocks can bridge the gap between FAIR Computational Workflows, interoperable reproducibility and for building canonical workflow descriptions to be used and described FAIRly across WfMSs.

The realisation of CWBBs can be achieved in many ways, not necessarily using the Python programming language together with RO-Crate as explored here. In particular, if the envisioned Canonical Workflow Frameworks for Research become established in multiple WfMSs with the use of FAIR Digital Objects, the different implementations will need to agree on object types, software packaging and metadata formats in order to reuse tools and provide interoperable reproducibility for canonical workflows.

Likewise, to build a meaningful collection of building blocks for a given research domain, a directed collaborative effort is needed to consistently wrap tools for a related set of WfMSs, chosen to target particular use cases (a family of canonical workflows).

For individual users, a library of Canonical Workflow Building Blocks simplifies many aspects of building pipelines, beyond the FAIR aspects and data compatibility across blocks. For instance, they can benefit from training of a CWBB family using Jupyter Notebooks, and then use this knowledge to utilise the same building blocks in a scalable HPC workflow with a CWL engine like Toil, knowing they will perform consistently thanks to the use of containers.

While we have demonstrated CWBB in the biomedical domain, this approach is generally applicable to a wide range of sciences that execute pipelines of multiple file-based command line tools, however it may be harder to achieve with more algebraic “in memory” types of computational workflows, where steps could be challenging to containerize and distinguish as separate block.

We admit that biomolecular research is quite a homogenous field with respect to computational analyses and now becoming relatively mature in terms of tool composability in workflows, building on the experiences of the “FAIR pioneers” in the field of bioinformatics. Other fields, such as social sciences or ecology, can have a wider variety of methods and computational tools, often with human interactions, and may have to adapt the software to be workflow-ready [37] before using them as Canonical Workflow Building Blocks. Domains adapting CWBB approach (or workflow systems in general) should take note of the great benefits

of hosting collaborative events where developers meet each other and their potential users, demonstrated in our field with events such WorkflowsRI [39] and Biohackathons [40].

The Common Workflow Language shows promise as a general canonical workflow building blocks mechanism: gathering execution details of tools along with their metadata and references, augmented with [abstract workflows](#) to represent canonical workflows. However, this would need further work to implement our CWBB recommendations in full. Future work for the Canonical Workflow Building Blocks concept includes formalising and automating publication practises, to make individual blocks available as FAIR Digital Objects on their own or as part of an aggregate collection like RO-Crate.

ACKNOWLEDGEMENTS

This work has been done as part of the BioExcel CoE (<https://www.bioexcel.eu/>), a project funded by the European Union contracts H2020-INFRAEDI-02-2018 823830, and H2020-EINFRA-2015-1 675728. Additional work is funded through EOSC-Life (<https://www.eosc-life.eu/>) contract H2020-INFRAEOSC-2018-2 824087, and ELIXIR-CONVERGE (<https://elixir-europe.org/>) contract H2020-INFRADEV-2019-2 871075.

AUTHOR CONTRIBUTIONS

Stian Soiland-Reyes (soiland-reyes@manchester.ac.uk): Conceptualization, Funding acquisition, Investigation, Methodology, Project administration, Supervision, Writing—original draft, Writing—review & editing

Genís Bayarri (genis.bayarri@irbbarcelona.org): Software, Software Documentation

Pau Andrio (andriopau@gmail.com): Methodology, Software, Validation, Software Documentation

Robin Long (r.long@lancaster.ac.uk): Software, Software Documentation

Douglas Lowe (douglas.lowe@manchester.ac.uk): Software, Software Documentation

Ania Niewielska (aniewielska@ebi.ac.uk): Methodology, Resources, Software

Adam Hospital (adam.hospital@irbbarcelona.org): Methodology, Project administration, Resources, Software, Validation, Visualization, Writing—original draft, Writing—review & editing

Paul Groth (p.t.groth@uva.nl): Supervision, Writing—review & editing

The authors would also like to acknowledge contributions from: Felix Amaladoss, Cibin Sadasivan Baby, Finn Bacall, Rosa M. Badia, Sarah Butcher, Gerard Capes, Michael R. Crusoe, Alberto Eusebi, Carole Goble, Josep Lluís Gelpí, Modesto Orozco, and Geoff Williams.

REFERENCES

- [1] Stodden, V., et al.: Enhancing reproducibility for computational methods. *Science* 354(6317), 1240–1241 (2016)
- [2] Leipzig, J., et al.: The role of metadata in reproducible computational research. *Patterns* 2(9), 100322 (2021)
- [3] Katz, D.S., et al.: A fresh look at FAIR for research software. arXiv preprint arXiv:2101.10883 (2021)
- [4] Möller, S., et al.: Robust cross-platform workflows: How technical and scientific communities collaborate to develop, test and share best practices for data analysis. *Data Science and Engineering* 2, 232–244 (2017)
- [5] Cohen-Boulakia, S., et al.: Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. *Future Generation Computer Systems* 75, 284–298 (2017)
- [6] Grüning, B., et al.: Practical computational reproducibility in the life sciences. *Cell Systems* 6(6), 631–635 (2018)
- [7] Lamprecht, A.-L., et al.: Towards FAIR principles for research software. *Data Science* 3(1), 37–59 (2020)
- [8] De Smedt, K., Koureas, D., Wittenburg, P.: FAIR digital objects for science: From data pieces to actionable knowledge units. *Publications* 8(2), 21 (2020)
- [9] Goble, C., et al.: (2020): FAIR Computational Workflows. *Data Intelligence* 2(1), 108–121 (2020)
- [10] Andrio, P., et al.: BioExcel building blocks, a software library for interoperable biomolecular simulation workflows. *Scientific Data* 6, 169 (2019)
- [11] Ison, J., et al.: EDAM: An ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics* 29(10), 1325–1332 (2013)
- [12] Hospital, A., et al.: BioExcel-2 Deliverable 2.3—First release of demonstration workflows (2020). Available at: <https://doi.org/10.5281/zenodo.4540432>. Accessed 26 November 2021
- [13] Kluyver, T., et al.: (2016): Jupyter notebooks—a publishing format for reproducible computational workflows. In: Loizides, F., Schmidt, B. (eds.) *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pp 87–90. Available at: <https://doi.org/10.3233/978-1-61499-649-1-87>. Accessed 26 November 2021
- [14] Beg, M., et al.: Using Jupyter for reproducible scientific workflows. *Computing in Science & Engineering* 23(2), 36–46 (2021)
- [15] Jupyter Project, Bussonnier, M., et al: Binder 2.0—Reproducible, interactive, sharable environments for science at scale. In: Proceedings of the 17th Python in Science Conference (SciPy 2018), pp. 113–120 (2018)
- [16] Grüning, B., et al.: Bioconda: Sustainable and comprehensive software distribution for the life sciences. *Nature Methods* 15, 475–476 (2018)
- [17] Niewielska, A., Butcher, S., Westermaier, Y.: BioExcel-2 Deliverable 2.5—Provision of a workflow environment at BioExcel portal. Available at: <https://doi.org/10.5281/zenodo.4916060>. Accessed 26 November 2021
- [18] Afgan, E., et al.: The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Research* 46(W1), W537–W544 (2018)
- [19] Crusoe, M.R., et al.: Methods included: Standardizing computational reuse and portability with the common workflow language. *Communication of the ACM* (in press). Available at: <https://doi.org/10.1145/3486897>. Accessed 26 November 2021
- [20] Tejedor, E., et al.: PyCOMPSs: Parallel computational workflows in Python. *The International Journal of High Performance Computing Applications* 31(1), 66–82 (2017)
- [21] Fillbrunn, A., et al.: KNIME for reproducible cross-domain analysis of life science data. *Journal of Biotechnology* 261, 149–156 (2017)
- [22] Lowe, D.: Protein MD setup tutorial using BioExcel building blocks (biobb) in Galaxy. WorkflowHub. Workflow (Galaxy). Available at: <https://doi.org/10.48546/workflowhub.workflow.194.1>. Accessed 26 November 2021

- [23] Hospital, A.: Protein MD setup tutorial using BioExcel building blocks (biobb) in KNIME. WorkflowHub. Workflow (KNIME). Available at: <https://doi.org/10.48546/workflowhub.workflow.201.1>. Accessed 26 November 2021
- [24] Bayarri, G., Long, R.: Protein MD setup tutorial using BioExcel building blocks (biobb) in CWL. WorkflowHub. Workflow (CWL). Available at: <https://doi.org/10.48546/workflowhub.workflow.29.3>. Accessed 26 November 2021
- [25] Bayarri, G., Hospital, A., Lowe, D.: (2021): Protein MD setup tutorial using BioExcel building blocks (biobb) in Jupyter Notebook. WorkflowHub. Workflow (Jupyter Notebook). Available at: <https://doi.org/10.48546/workflowhub.workflow.120.2>. Accessed 26 November 2021
- [26] Hospital, A., Andrio, P.: Protein MD setup HPC tutorial using BioExcel building blocks (biobb) in PyCOMPSs. WorkflowHub. Workflow (PyCOMPSs). Available at: <https://doi.org/10.48546/workflowhub.workflow.200.1>. Accessed 26 November 2021
- [27] The BioMoby Consortium: Interoperability with Moby 1.0—It's better than sharing your toothbrush! *Briefings in Bioinformatics* 9(3), 220–231 (2008)
- [28] Saltz, J., et al.: caGrid: Design and implementation of the core architecture of the cancer biomedical informatics grid. *Bioinformatics* 22(15), 1910–1916 (2006)
- [29] Garijo, D., Gil, Y.: A new approach for publishing workflows. In: Proceedings of the 6th Workshop on Workflows in Support of Large-Scale Science—WORKS '11, pp. 47–56 (2011)
- [30] Garijo, D., et al.: Common motifs in scientific workflows: An empirical analysis. *Future generation computer systems* 36, 338–351 (2014)
- [31] De Giovanni, R., et al.: ENM components: A new set of Web service - based workflow components for ecological niche modelling. *Ecography* 39(4), 376–383 (2016)
- [32] Blankenberg, D., et al.: Dissemination of scientific software with Galaxy ToolShed. *Genome Biology* 15: Article No. 403 (2014)
- [33] Vivian, J., et al.: Toil enables reproducible, open source, big biomedical data analyses. *Nature Biotechnology* 35 pp 314–316 (2017)
- [34] Soiland-Reyes, S., et al.: Packaging research artefacts with RO-Crate. *Data Science* (in press). Available at: <https://doi.org/10.3233/DS-210053>. Accessed 26 November 2021
- [35] Ison, J., et al.: biotoolsSchema: A formalized schema for bioinformatics software description. *GigaScience*, 10(1):giaa157 (2021)
- [36] The CWFR Group, Hardisty, A., (ed.), Wittenburg, P., (ed.): CWFR position paper. OSF, January 6, 2021. Available at: <https://osf.io/3rekv/>. Accessed 26 November 2021
- [37] Brack, P., et al.: 10 simple rules for making a software tool workflow-ready. *PLOS Computational Biology* 18(3), e1009823 (in press). Available at: <https://doi.org/10.1371/journal.pcbi.1009823>. Accessed 26 November 2021
- [38] McMurry, J.A., et al.: Identifiers for the 21st century: How to design, provision, and reuse identifiers to maximize utility and impact of life science data. *PLOS Biology* 15(6): e2001414 (2017)
- [39] Ferreira da Silva, R., et al.: A community roadmap for scientific workflows research and development. *arXiv preprint arXiv: 2110.02168* (2021)
- [40] Garcia, L., et al.: Ten simple rules to run a successful BioHackathon. *PLOS Computational Biology* 16(5): e1007808 (2020)

AUTHOR BIOGRAPHY

Stian Soiland-Reyes is a Technical Architect in the eScience Lab at The University of Manchester, and a Ph.D. Candidate in the INDE Lab at University of Amsterdam. Since 2006 he has worked as a software engineer and researcher focusing on reproducibility, scientific workflows, interoperability, linked data, metadata and open science. He is a persistent advocate of Open Scholarly Communication, and is on the leadership teams of Common Workflow Language and BioCompute Objects. He contributed to the W3C provenance model PROV-O and multiple Linked Data initiatives. He co-created the Research Object model and is co-chair of the Research Object Crate community.

ORCID: 0000-0001-9842-9718



Genís Bayarri is a Full Stack Developer working as a Web Developer and Software Research Engineer in the Molecular Modeling and Bioinformatics group of the Institute for Research in Biomedicine in Barcelona (IRB Barcelona). He graduated in Multimedia Engineering, and has a broad experience in the private sector. He joined the IRB Barcelona in 2016 and since then he has participated in the European projects BioExcel Centre of Excellence and Multiscale Complex Genomics. He has also developed a set of public Web servers, databases and applications related to molecular dynamics, specially in the 3D representation field.

ORCID: 0000-0003-0513-0288



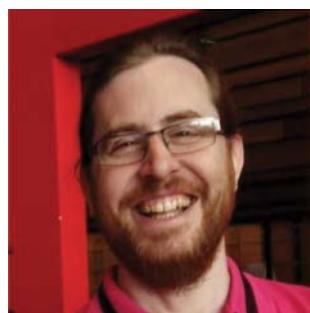
Pau Andrio is a Research Software Engineer (RSA) working for the Spanish National Bioinformatics Institute (INB) at the Barcelona Supercomputing Center (BSC). He graduated from the Universitat Jaume I (UJI) with a Bachelor's degree in Computer Sciences and a Master's degree in Biomedicine from Universitat de Barcelona (UB) in 2012. Since 2010, he has been involved in several projects in the field of bioinformatics, structural biology and molecular dynamics simulation, also contributing to two main EU projects: Scalalife and BioExcel. His main interest is to provide the life sciences research community; user-friendly, reproducible and scalable software. One of the results of BioExcel, and the main project he is currently contributing to, is the BioBB (BioExcel Building Blocs) library.

ORCID: 0000-0003-2116-3880



Robin Long is a Research Software Engineer at the Data Science Institute of Lancaster University, where he is providing training in research computing and software skills whilst supporting various research software projects. Previously he worked as a Research Software Engineer at The University of Manchester where he developed and improved software tools used for computational biomolecular research and analysis workflows in BioExcel. He is a 2017 Fellow of the Software Sustainability Institute.

ORCID: 0000-0003-2249-645X



Douglas Lowe is a Research Software Engineer working in the Research-IT department, and eScience Lab at the University of Manchester. He gained his Ph.D. degree in Environmental Sciences at Lancaster University in 2004, and has subsequently worked as a Research Associate at the University of Manchester, focusing on studying air quality around the world using large-scale, open source, atmospheric models. He moved to the Research-IT department in 2019, and has been working with the eScience Lab since 2020, where he has been working on developing scientific workflows in Common Workflow Language using the BioExcel Building Blocks library.

ORCID: 0000-0002-1248-3594



Ania Niewielska is a Lead Software Engineer in the Software Development & Operations team, part of the Technical Services Cluster at EMBL-EBI. She holds a Master's degree in Computer Science from the Silesian University of Technology in Poland. Joining EMBL-EBI in 2017, she brought with her a passion for human-centred software design, as well as extensive experience of system integration and API design from the private sector. More recently she has developed an interest in cloud computing and container orchestration, and their application in scientific workflows. Ania co-leads the development of GA4GH Task Execution Service API. Her team in EMBL-EBI develops and maintains the BioExcel Cloud Portal and its BinderHub component.

ORCID: 0000-0003-0989-3389



Adam Hospital is a Research Software Engineer working as a Research Associate in the Molecular Modeling and Bioinformatics group of the Institute for Research in Biomedicine in Barcelona, Spain. Graduated in Computational Science, he received his PhD in Biotechnology for the University of Barcelona in 2013. Since then, he has been working in computational biomolecular simulation workflows in the Spanish National Institute of Bioinformatics (INB, ELIXIR-ES), and is leading the workflows team of the BioExcel Center of Excellence. Interested in making scientific software, workflows and tools publicly available to the scientific community, he has coordinated and participated in the development of a set of public web servers and databases related to macromolecular structure flexibility.

ORCID: 0000-0002-8291-8071



Paul Groth is Professor of Algorithmic Data Science at the University of Amsterdam where he leads the Intelligent Data Engineering Lab (INDElab) and is scientific director of the university's Data Science Centre. He holds a Ph.D. in Computer Science from the University of Southampton (2007) and has done research at the University of Southern California, the Vrije Universiteit Amsterdam and Elsevier Labs. His research focuses on intelligent systems for dealing with large amounts of diverse contextualized knowledge with a particular focus on Web and science applications. This includes research in data provenance, data integration and knowledge sharing.

ORCID: 0000-0003-0183-6910